Monte Carlo Tree Search with Adaptive Simulation

a Case Study on Weighted Vertex Coloring Problem

Cyril Grelier
Olivier Goudet
Jin-Kao Hao

EVOCOP 2023

Sommaire

Introduction

WVCP - Weighted Vertex Coloring Problem

Given a graph $G = (V, E)$, V the vertices, E the edges of the graph and $w(v)$ the weight of v for $v \in V$

The objective is to find a legal coloring s (two connected vertices can't share the same color) that minimizes the sum of the heaviest vertices of each k colors.

$$score = F(s) = \sum_{i=1}^{k} max_{v \in V_i} w(v)$$

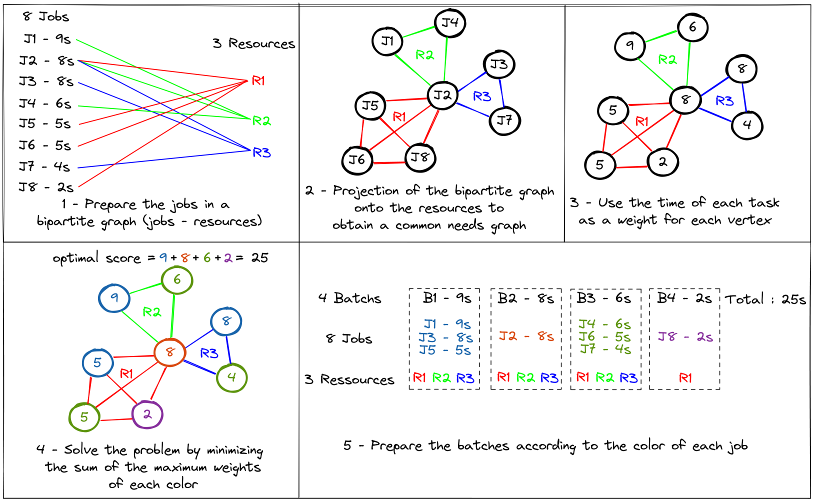
Applications :

- Scheduling on a Batch Machine with Job Compatibilities
- Traffic assignment in communication satellites
- Matrix Decomposition Problem

NP-hard problem

demo

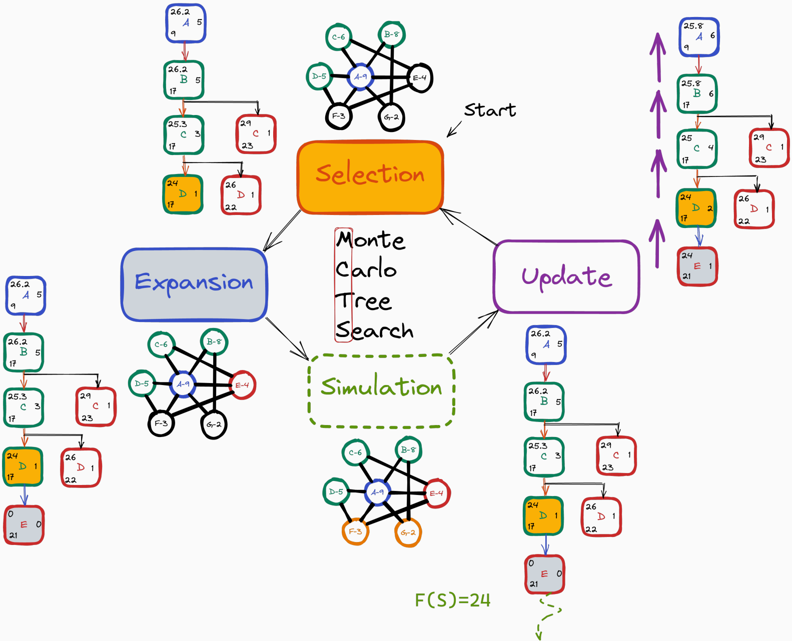
Exemple of application



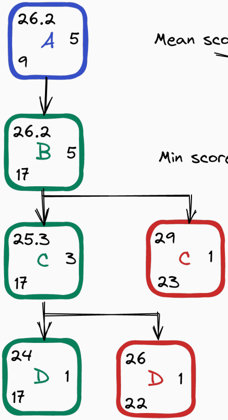
State of the art - WVCP

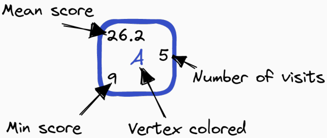
- **R-GRASP** : reactive GRASP, iterated greedy algorithm with local search.
  Prais, M., Ribeiro, C.C. , 2000.
  Reactive GRASP : An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment.
- **2-Phase** : Phase 1 : generation of independent sets, phase 2 : optimization.
  Malaguti, E., Monaci, M., Toth, P., 2009.
  Models and heuristic algorithms for a weighted vertex coloring problem.
- **MWSS** : mixed integer linear programming.
  Cornaz, D., Furini, F., Malaguti, E., 2017.
  Solving vertex coloring problems as maximum weight stable set problems.
- **AFISA** : tabu search with coefficient to manage GCP and WVCP.
  Sun, W., Hao, J.-K., Lai, X., Wu, Q., 2018.
  Adaptive feasible and infeasible tabu search for weighted vertex coloring.
- **RedLS** : reduction and local search with weight management of the edges.
  Wang, Y., Cai, S., Pan, S., Li, X., Yin, M., 2020.
  Reduction and Local Search for Weighted Graph Coloring Problem.
- **ILS-TS** : reduction and iterated local search with grenade operator.
  Nogueira, B., Tavares, E., Maciel, P., 2021.
  Iterated local search with tabu search for the weighted vertex coloring problem.
- **DLMCOL** : memetic algorithm with deep learning for the crossover selection.
  Goudet, O., Grelier, C., Hao, J.-K., 2021.
  A deep learning guided memetic framework for graph coloring problems.
- **MCTS** : monte carlo tree search to find good initialisation for the local search.
  Grelier, C., Goudet, O., Hao, J.-K., 2022.
  On Monte Carlo Tree Search for Weighted Vertex Coloring.

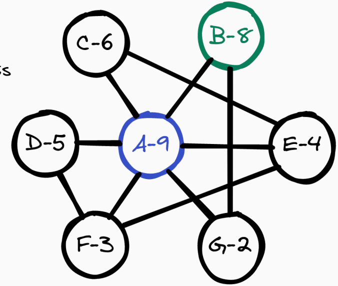
Proposed algorithm


Start
Selection
Expansion
Simulation
Update
Monte
Carlo
Tree
Search
F(S)=24

Tree

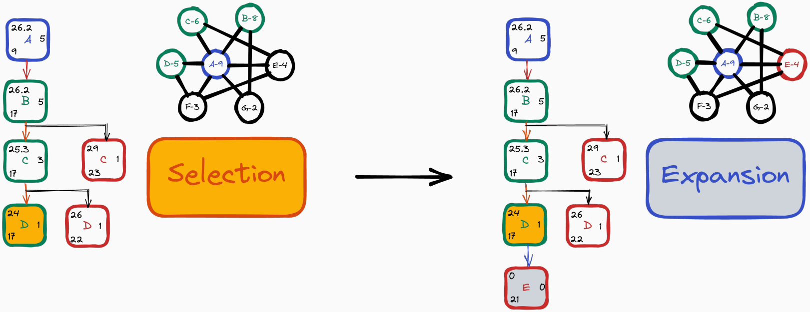

26.2
A 5
9
26.2
B 5
17
25.3
C 3
17
29
C 1
23
24
D 1
17
26
D 1
22


Mean score
26.2
A 5
9
Number of visits
Min score
Vertex colored


C-6
B-8
D-5
A-9
E-4
F-3
G-2

Graph

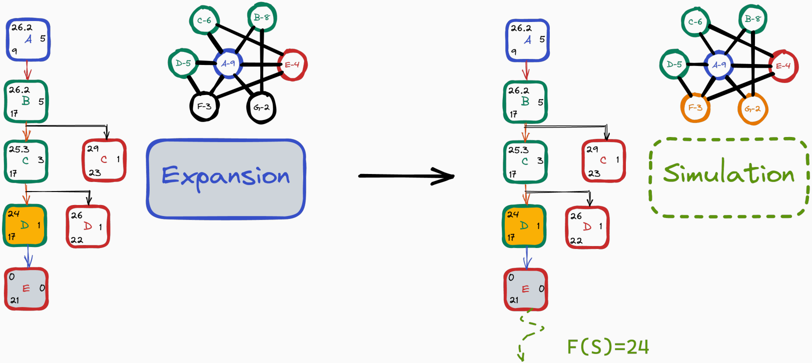
Monte Carlo Tree Search



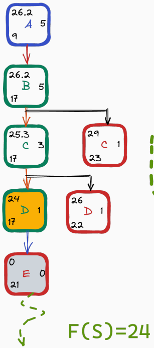
$$normalized\_score(C^i_{t+1}) = \frac{rank(C^i_{t+1})}{\sum_{i=1}^{l} rank(C^i_{t+1})}$$

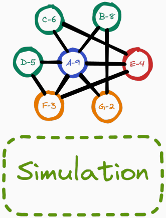
$$\text{(UCB)}\ normalized\_score(C^i_{t+1}) + c \times \sqrt{\frac{2*ln(nb\_visits(C_t))}{nb\_visits(C^i_{t+1})}}$$

Monte Carlo Tree Search



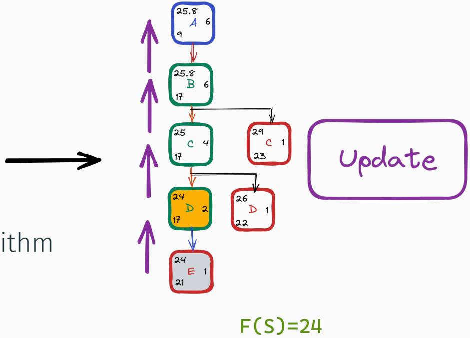
Monte Carlo Tree Search



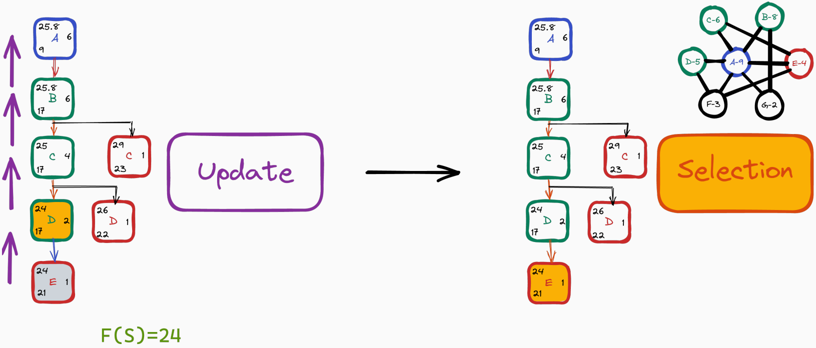


1- Greedy Algorithm

2- Local Search



Monte Carlo Tree Search



Which Local Search operators ?

- **AFISA** : illegal search space, one move operator, coefficient varies to come back to a legal solution
- **RedLS** : illegal search space, one move operator, weights on conflicting edges to come back to a legal solution and force heaviest vertices of color to move
- **ILSTS** : partial legal search space, one move and grenade operator, force heaviest vertices of multiple colors to move
- **TabuWeight (TW)** : legal search space, one move operator, inspired from TabuCol

Results

1 point for the method on the row if the difference on the mean score on the 20 runs is significantly better (non-parametric Wilcoxon signed-rank test with a p-value ≤ 0.001)

/188 instances	AFISA	MCTS+AFISA	TW	MCTS+TW	RedLS	MCTS+RedLS	ILSTS	MCTS+ILSTS	# BKS	# Best Score	# Best Mean
AFISA	-	35	**48**	16	**56**	0	9	13	115	115	49
MCTS+AFISA	**40**	-	**72**	10	**86**	0	1	1	116	116	98
TW	39	38	-	26	47	2	16	23	98	98	52
MCTS+TW	**68**	**71**	**78**	-	**99**	5	8	17	129	129	98
RedLS	44	38	47	29	-	13	23	25	112	130	46
MCTS+RedLS	**102**	**79**	**105**	**68**	**102**	-	**42**	**43**	**153**	**162**	**144**
ILSTS	**87**	**80**	**86**	**56**	**83**	13	-	**25**	**151**	**151**	**141**
MCTS+ILSTS	**82**	**78**	**83**	**48**	**81**	11	0	-	**148**	**148**	**141**

Monte Carlo Tree Search with adaptive simulation

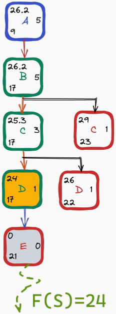
Why ?

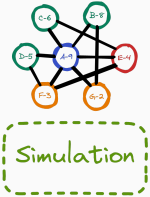
- **No best LS** : No Local Search dominates the others
- **Adapt** : Choose the right operator without previous knowledge

How ? Hyper-heuristics !

- **Selection criteria** : train a criterion to choose operators
- **Reward** : use the score of the solution after local search
- **Sliding window** : use a normalization of the scores during the *ws* last iterations

Monte Carlo Tree Search with adaptive simulation





Before :
1 – Greedy algorithm to complete the solution
2 – Apply a Local Search

Now :
1 – Greedy algorithm to complete the solution
2 – Criteria to choose the LS operator
3 – Apply the LS
4 – Update criteria with F(solution after LS)

Monte Carlo Tree Search with adaptive simulation

Which selection criteria ?

- **Random** : Random selection
- **Roulette Wheel** [1] : Fair selection depending on the results

$$proba[o] = p_{min} + (1 - n_o * p_{min}) * \frac{r[o]}{\sum r}$$

- **Pursuit** : Unfair selection in favor of the best operator (b)

$$\begin{cases} proba[b] = proba[b] + \beta(p\_max - proba[b]) \\ proba[o] = proba[o] + \beta(p\_min - proba[o]) \end{cases}$$

- **UCB** : Favouring the best operators while encouraging exploration

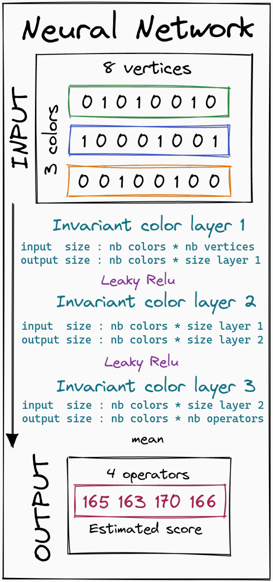
$$score[o] = r[o] + c * \sqrt{2 * \frac{\log(\sum visits)}{visits[o]}}$$

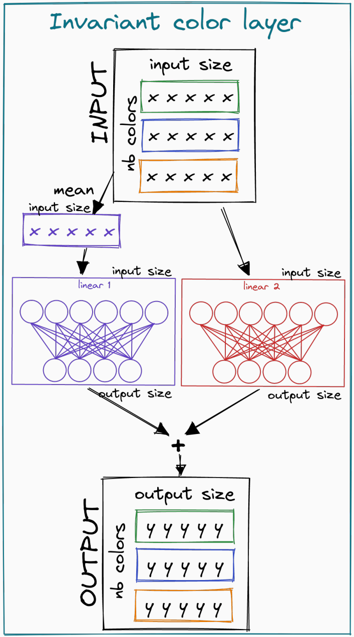
- **NN** : Recommendations of a Neural Network using deep sets [2] on a raw solution

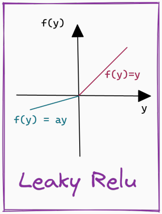
---

1. A. Goëffon et al. Simulating non-stationary operators in search algorithms. 2016
2. M. Zaheer et al. Deep Sets. 2018

Neural network – Deep sets

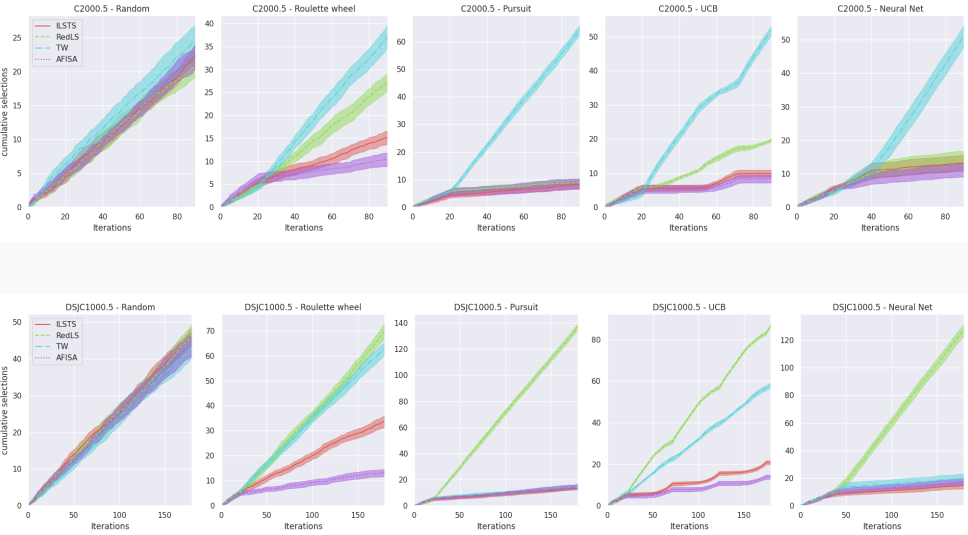




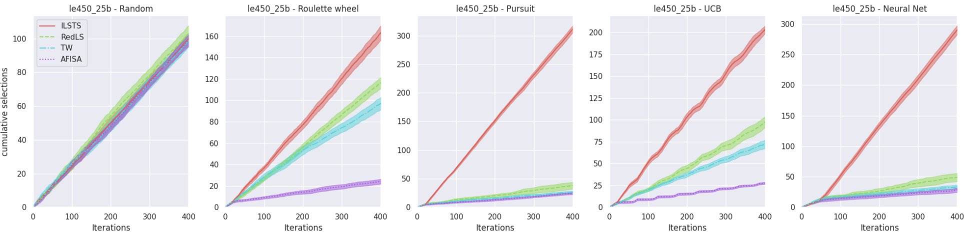


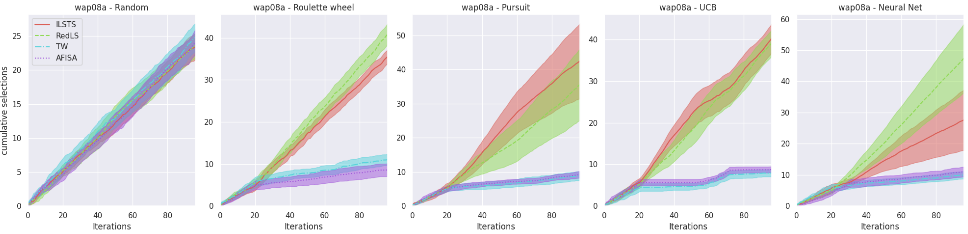
Experimentation

Operator selection



Operator selection





Results

1 point for the method on the row if the difference on the mean score on the 20 runs is significantly better (non-parametric Wilcoxon signed-rank test with a p-value ≤ 0.001)

/188 instances	AFISA	MCTS+AFISA	TW	MCTS+TW	RedLS	MCTS+RedLS	ILSTS	MCTS+ILSTS	Random	Roulette Wheel	Pursuit	UCB	Neural network	# BKS	# Best Score	# Best Mean
AFISA	-	35	**48**	16	**56**	0	9	13	0	0	0	0	0	115	115	49
MCTS+AFISA	**40**	-	**72**	10	**86**	0	1	1	0	0	0	0	0	116	116	98
TW	39	38	-	26	47	2	16	23	2	2	2	2	2	98	98	52
MCTS+TW	**68**	**71**	**78**	-	**99**	5	8	17	0	0	0	0	0	129	129	98
RedLS	44	38	47	29	-	13	23	25	15	15	15	15	14	112	130	46
MCTS+RedLS	**102**	**79**	**105**	**68**	**102**	-	**42**	**43**	**21**	7	6	4	4	153	**162**	144
ILSTS	**87**	**80**	**86**	**56**	**83**	13	-	**25**	8	5	1	2	4	151	151	141
MCTS+ILSTS	**82**	**78**	**83**	**48**	**81**	11	0	-	5	2	0	1	2	148	148	141
Random	**103**	**81**	**103**	**72**	**98**	12	**34**	**39**	-	0	0	0	1	154	158	139
Roulette Wheel	**103**	**81**	**104**	**73**	**100**	**13**	**35**	**40**	**4**	-	**1**	0	1	155	160	142
Pursuit	**103**	**81**	**105**	**75**	**101**	**14**	**35**	**40**	**11**	0	-	1	**2**	**156**	**163**	**155**
UCB	**103**	**81**	**104**	**75**	**100**	**13**	**36**	**40**	**4**	0	1	-	**1**	**156**	160	**146**
NN	**103**	**81**	**104**	**75**	**101**	**13**	**35**	**41**	**9**	**2**	0	0	-	**157**	**161**	**152**

Complete results and source code :
https://github.com/Cyril-Grelier/gc_wvcp_adaptive_mcts

Conclusion

Conclusion

MCTS with adaptive simulation for the weighted vertex coloring problem :

- highest number of BKS, good rank among other methods, and provide regularity in the results
- No change of operator during the search
  - usually one(/two) domminant operator for one instance
  - no complementarity in using different operators

Thank you for your attention!

Questions?

`https://cyril-grelier.github.io/`

Selection criteria - Criteria based on probabilities

Random selection with bias[3]

- **Fixed random** (with bias or not) :
  Same probability during all the search
- **Adaptive roulette wheel** :
  The more an operator *o* reaches good scores, the more it will be selected

$$proba[o] = p_{min} + (1 - n_o * p_{min}) * \frac{r[o]}{\sum r}$$

n_O : number of operators
r : vector of the normalization of the sum of the obtained rewards during the last *ws* generations for each operators
p_{min} : probability minimum (0.05)

- **Adaptive pursuit** :
  The operator with the best results (b) get far more chances to be selected

$$\begin{cases} proba[b] = proba[b] + \beta(p\_max - proba[b]) \\ proba[o] = proba[o] + \beta(p\_min - proba[o]) \end{cases}$$

p_{max} : probability maximum ($= 1 - (n_O - 1) * p_{min}$)

3. A. Goëffon, F. Lardeux, and F. Saubion. Simulating non-stationary operators in search algorithms. 2016

Selection criteria - Criteria based on multi-armed bandit

Selection of the most profitable (best score)[2]

- UCB :
  The better scores an operator gets, the more likely it is to be selected (exploitation) but the less an operator is selected, the more likely it is to be selected (exploration)

$$score[o] = r[o] + c * \sqrt{2 * \frac{\log(\sum visits)}{visits[o]}}$$

r : vector of the normalization of the sum of the obtained rewards during the last *ws* generations for each operators

visits : number of time each operator have been selected during the last *ws* generations

2. A. Goëffon, F. Lardeux, and F. Saubion. Simulating non-stationary operators in search algorithms. 2016

Selection criteria - Criteria based on a neural network

Selection of the best according to a neural network

- **Neural network** :
  Use a raw solution as input and predict the reward for each operator
  10% chance of random selection
  Use deep set [4]

---

4. M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep Sets. 2018